

Agent Observability

智能体可观测性

2026年5月22日 | AI Agent 知识分享 | 第2期

一、一句话概括

智能体可观测性是对Agent的思考轨迹、工具调用链、Token消耗和决策过程进行全链路追踪与可视化，让你能像调试Web应用一样「扒开」智能体的黑盒，定位它为什么答错、为什么卡住、为什么超支。

二、核心原理

>> 三大支柱：Tracing、Metrics、Logging

Agent可观测性围绕三个核心维度展开：Tracing（追踪）：记录Agent每一步的思考内容、工具调用输入输出和执行耗时，形成完整的调用链，是定位问题的核心手段；Metrics（指标）：聚合统计完成率、平均步数、Token消耗、工具调用成功率等量化数据；Logging（日志）：结构化存储Agent的推理过程和异常状态，支持事后回溯分析。三者构成可观测性的完整闭环。

>> 关键指标面板（Key Metrics Dashboard）

Agent可观测性需要监控的核心指标包括：完成率（Completion Rate） - Agent成功完成任务的百分比；平均步数（Avg Steps） - 完成任务所需的思考-行动循环次数；平均耗时（Avg Latency） - 从开始到完成的平均时间；Token效率（Token Efficiency） - 每步消耗Token与任务完成质量的关系；工具调用成功率（Tool Success Rate） - 工具调用成功的比例；回退率（Backtrack Rate） - Agent需要回退重新规划的频率。这些指标通过LangFuse/LangSmith等平台聚合展示。

>> 分布式追踪与Trace ID

在多Agent或微服务架构中，Agent A可能调用Agent B，B又调用外部API和数据库，形成复杂的调用链路。通过统一的Trace ID传递机制，可以将分散在不同组件中的执行记录关联起来，还原完整的执行路径。当前主流方案基于OpenTelemetry规范，Trace ID在进程间通过HTTP Header或消息队列元数据传递。Hermes Agent的session_search机制本质上也是一种Agent级的追踪能力。

核心要点：可观测性不仅是Debug工具，更是Agent治理的基础设施

Trace ID 是打通多Agent调用链的关键设计

Metrics 面板要兼顾「快速排查」和「趋势分析」两个场景

三、测试实战

测试一：Agent思考链完整性测试 (Trace Completeness)

操作：部署Agent应用，集成OpenTelemetry SDK，构造一个需要3步工具调用的任务（如「查询A城市的天气→对比B城市的温度→给出出行建议」），完成后检查Trace是否完整记录了所有3步思考过程和工具调用

工具：LangFuse (<https://langfuse.com>) 或 LangSmith (<https://smith.langchain.com>) 查看完整Trace

指标：Trace完整性评分（记录步数/实际步数×100%）、工具输入输出记录完整率、思考内容截断率

测试二：Token消耗追踪准确性测试

操作：在Agent执行过程中手动记录每个步骤的Token消耗，与可观测性平台展示的数据对比，验证Token统计是否准确（特别注意输入Prompt中的历史累积是否正确计入）

工具：LangFuse Token Usage面板 / OpenAI API用量控制台

指标：Token统计误差率（≤5%为合格）、输入输出Token分离准确率、历史累积Token重复计数率

测试三：异常场景告警测试 (Alert Accuracy)

操作：制造异常场景——让Agent连续失败、工具超时、Token超额，验证可观测性系统的告警机制能否正确触发，且不产生误报

工具：Grafana Alerting (<https://grafana.com/alerting>) 或 Prometheus + Alertmanager

指标：告警召回率（真实异常的检出率）、误报率（False Positive Rate）、告警平均延迟（从异常发生到告警触发的秒数）

测试四：Trace ID跨进程追踪测试

操作：设计一个多Agent协作任务（Agent A → Agent B → 第三方API），验证Trace ID能否正确跨进程传递，在可观测性平台上看到完整的端到端调用链

工具：OpenTelemetry Collector + Jaeger (<https://www.jaegertracing.io/>) 查看分布式Trace

指标：Trace ID传递成功率、跨进程调用链完整率、子Span正确归因率（Parent-Child关系准确度）

四、常见坑点

坑点1：日志数据量爆炸

Agent的每一步思考内容、工具调用输入输出会产生大量日志。一个10步的Agent任务可能产生数万Token的追踪数据，多Agent场景下更是几何级增长。解决方案：配置采样率（生产环境建议1:10~1:100采样）、设置日志保留策略（热数据保留7天+冷数据归档）、启用智能降采样（只记录关键决策步和异常步）。

坑点2：敏感信息泄露

工具调用的输入输出可能包含用户敏感数据（姓名、身份证号、业务数据等），如果直接在日志中明文记录将造成合规风险。必须实现脱敏处理——在记录前自动识别并遮盖敏感字段（正则匹配手机号/身份证号/PII模式），或在存储层加密。LangFuse等平台已内置PII脱敏功能，建议优先启用。

坑点3：分布式追踪中的数据一致性

在多Agent架构中，Trace ID的传递依赖HTTP Header或消息队列元数据，在异步回调场景下极易丢失。常见问题：Agent A异步调用了Agent B，但B的处理结果通过回调返回时丢失了Trace ID，导致AB两段Trace无法关联。建议实现「Trace ID强制传递」机制——所有跨服务调用在框架层自动注入Trace ID，而非依赖开发者手动传递。

五、延伸阅读

- LangFuse 开源Agent可观测性平台: <https://langfuse.com>
- LangSmith Agent Monitoring: <https://smith.langchain.com>
- OpenTelemetry Agent Tracing 规范: <https://opentelemetry.io/docs/specs/semconv/gen-ai/>
- Jaeger 分布式追踪: <https://www.jaegertracing.io/>
- Hermes Agent session_search 机制 (Agent级追踪实现) : <https://hermes-agent.nousresearch.com>